



AI'STORY

인공지능이 지키는 안전한 우리집

사회적 소외계층을 위한 화재감지 AIoT 플랫폼

2023 캡스톤디자인 졸업작품

팀 하루

최은서, 최아영, 이혜빈



INDEX

01 | 개요

- 프로젝트 추진배경
- 프로젝트의 필요성
- 프로젝트의 목적

02 | 시스템 분석 설계 과정

- SWOT & CROSS SWOT
- 유스케이스 다이어그램
- 아키텍처 설계

03 | 구현 및 기술

- 개발 환경 및 설명
- 센서 구현
- 인공지능 구현
- 인공지능 구현 시각화

04 | 기대효과

- 사회적 기대효과
- 경제적 기대효과
- 실현 가능성 및 활용 분야

05 | 확장성 및 개선 방향

06 | 시연 영상

01

개요

프로젝트 추진 배경

1. 올해 초 서울특별시의 빈민가인 구룡마을에서 대형 화재가 발생했다는 기사를 접함.
 - > 화재들이 대형화재로 번지는 이유에 대해 조사하게 됨.
 - > 여러 원인이 존재하나, 전통적으로 화재 발견과 신고가 늦거나 이로 인해 초기 진압이 늦어져 화재가 더 커지는 것이라는 지적을 보게 됨.
2. 사회적 소외계층이 집약되어 있는 곳은 주택 밀집도가 높기 때문에 소방력이 침투하기 어려워 초기 진압과 화재가 발생한 정확한 위치를 파악하기 어렵고, 집의 건축 자재가 불이 붙기 쉬운 소재로 지어진 사실을 알게됨.
3. 구룡마을 화재의 원인은 전기화재라고 추정할 뿐 정확한 화재 원인을 특정할 수 없다는 기사를 봄.
 - > 이에 전기화재에 대해 조사하다 아크 즉, 스파크를 통한 전기적 화재와 관련된 논문을 찾아보게 되었고, 직렬아크는 노화된 전선으로 인한 화재의 원인이라는 사실을 알게됨.
 - > 또한 소방청 국가화재정보센터가 제공하는 국가화재통계의 화재 원인 중 '원인을 알 수 없지만 이것으로 인해 발생했다고 추정'하는 미확인 단락의 경우 전기화재에 유일하게 존재한다는 사실을 알게되었다.
4. 위의 근거들로 노후화된 기기나 전선이 많은 소외계층이 집약된 곳에서 직렬아크로 인한 전기적 화재가 많이 발생할 수 있을 것이라는 가설을 세우게 되었고, 화재 피해를 최소화 할 수 있는 AIoT플랫폼을 고안하게 되었음.

화재가 발생한 구룡마을 모습



화재 원인 조사

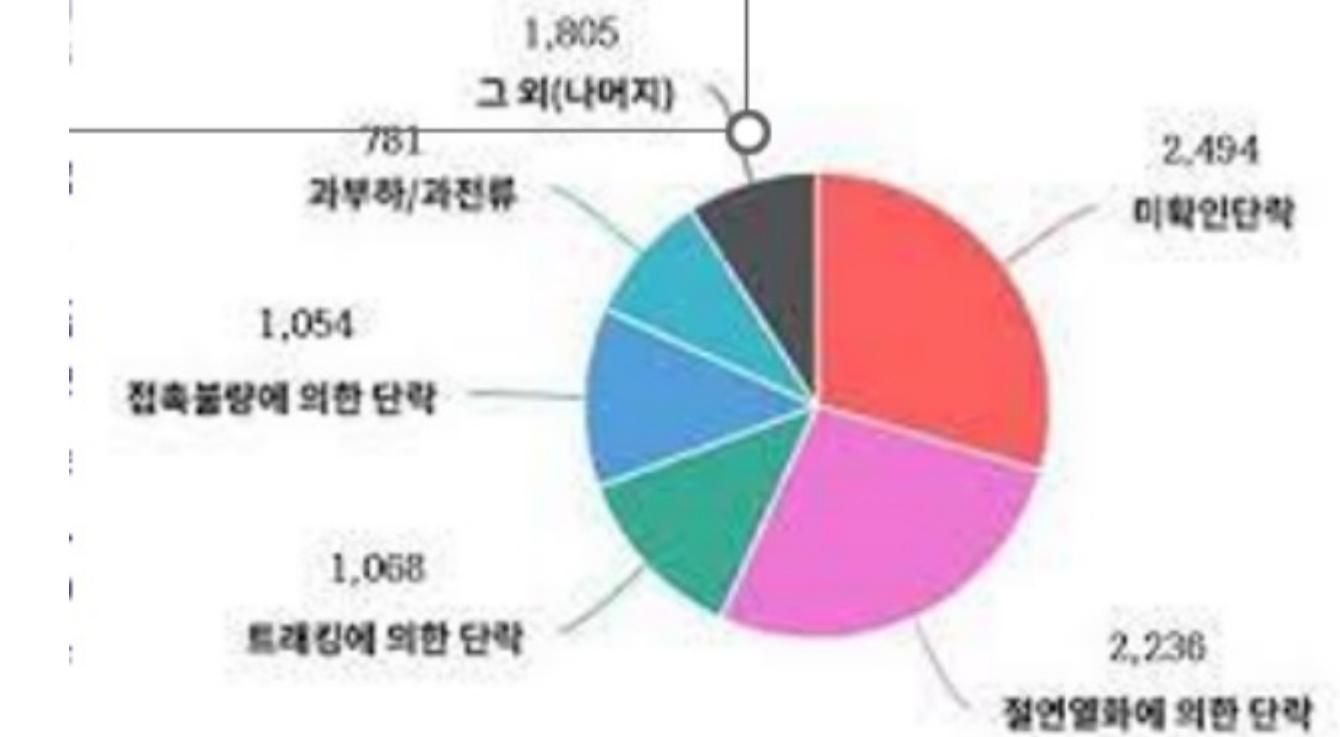


누전에 의한 발화 : 3.9%

스파크에 의한 발화 : 78%

아크에 의한 칼퇴진수는 전기화재 7,760건 중 6,057건으로 78%정도

[발화요인 - 전기적 요인 - 전체] 화재건수 비교



프로젝트 필요성

- 노후화된 기기나 전선이 많은 소외계층이 집약된 곳은 집에 불이 잘 옮겨붙고 타는 자재들로 건축되어 있음.
- 이러한 곳에서 전기적 화재가 일어난다면 순식간에 화재가 번지는 것은 물론이고 집 사이의 거리가 가까워 정 확한 화재 근원지를 발견하기 어렵고, 좁은 통로로 인해 소방력이 진입하기 힘듦.



- 사회적 소외계층의 안전을 위한 화재 예측 및 감지 AIoT플랫폼을 사용한다면 사람이 화재를 발견하기까지 신 고를 기다리는 것이 아닌 **인공지능이 화재를 감지한 즉시 소방청에 신고**를 함과 동시에 **화재 발생 주소지를 전 달하여 소방력이 빠르게 진입할 수 있음.**
- 원인을 특정하기도 힘들고, 관련된 데이터를 얻는 것이 힘든 **전기적 화재에 대한 데이터를 쌓을 수 있으며,** 더 나아가 지역별, 계절별, 습도별 등 **다양한 범주에 따른 화재 데이터를 수집하고 제공**할 수 있기 때문에 **화재 예 방 또는 화재 진압 매뉴얼의 발전**에 데이터적인 도움을 줄 수 있다는 이점을 가지고 있음.

주택 밀집도에 대한 기사

이종석 강남소방서 소방위는 “전기 또는 기계적이거나 부주의 등 다양한 요인 중에서 화기를 잘못 사용하거나 전자제품이 합선·누전되면서 발생하는 화재가 겨울철에는 가장 많다”고 설명했다. 이 소방위는 “특히 구룡마을은 건물이 낡은 데다 주택 밀집도가 높고 소방통로 확보가 어려워 작은 불이 나면 순식간에 옆 건물로 번져 여러 채가 소실될 수 있는 취약 지역”이라며 “이번 화재 원인은 조사 중”이라고 설명했다.

건축 자재에 대한 기사

구룡마을은 '떡솜'으로 불리는 단열재와 비닐·합판·스티로폼 등 불이 붙기 쉬운 소재로 지은 가건물이 밀집해 화재에 취약한 구조다.

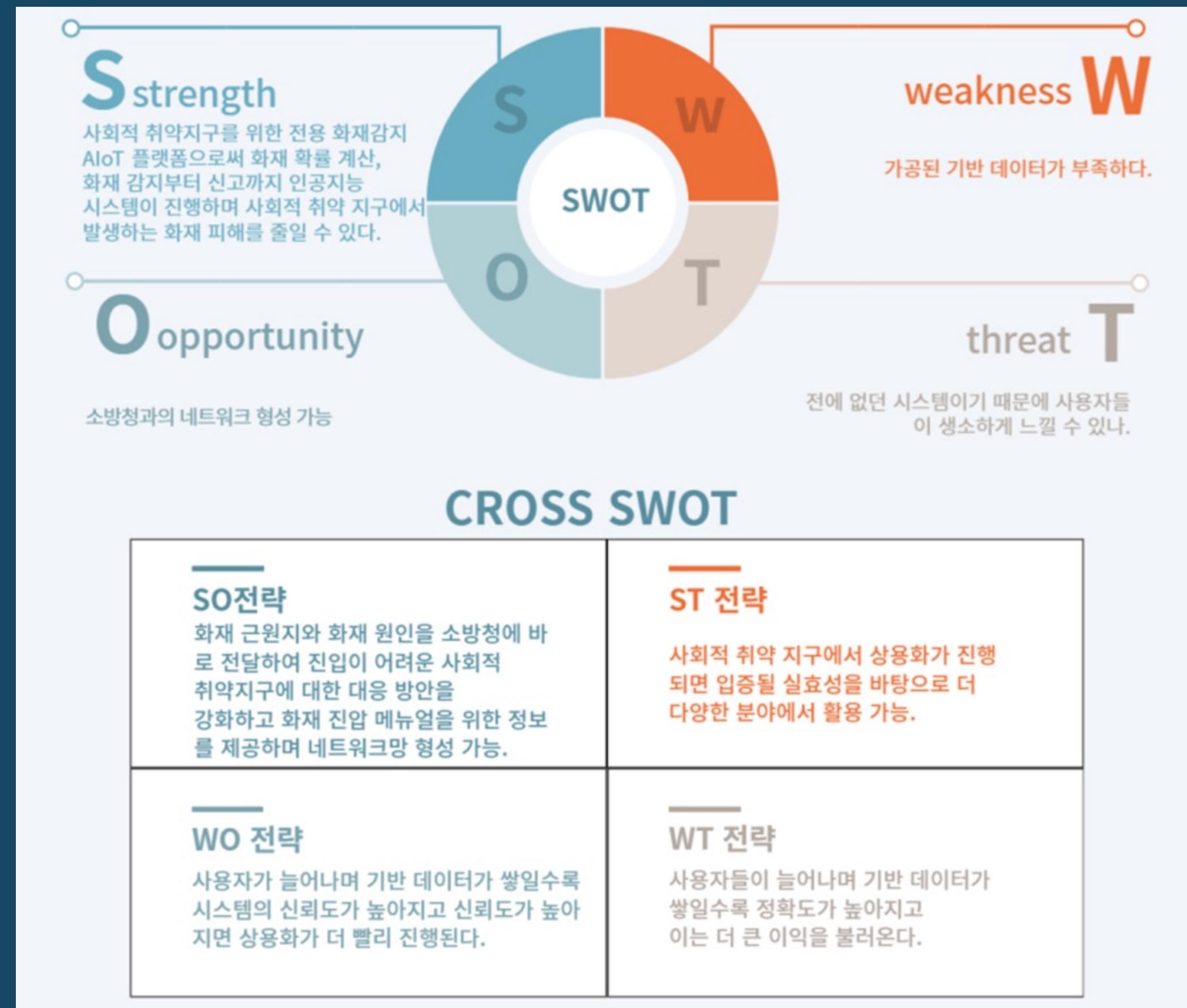
프로젝트 목적

- 스파크나 순간적인 고온을 감지할 경우 '이상치 발견 알림'을 띄우고, 기존 화재 데이터를 기반으로 온습도, 계절 등의 데이터로 화재 발생 확률을 계산하며, 실질적인 화재를 감지할 경우 즉시 소방서에 신고를 해 재산 피해와 인명 피해를 줄이는 것이 이 AIoT플랫폼의 가장 중요한 역할이다.
- 사회적 소외계층의 안전을 위한 화재 예측 및 감지 AIoT 플랫폼 AI'story는 그들의 삶의 터전을 보호하는데 목표를 두고 있다.
- 쌓인 빅데이터를 공공기관에 제공하여 빈민가, 판자촌 등 소외계층이 집약된 즉, 좁은 골목 탓에 소방력이 침투하기 어려운 곳들에 대한 화재 진압, 화재 진압을 위한 진입 등의 매뉴얼을 강화시켜 부득이한 화재로 인한 피해를 최소화한다는 거시적인 목표를 가지고 있다.

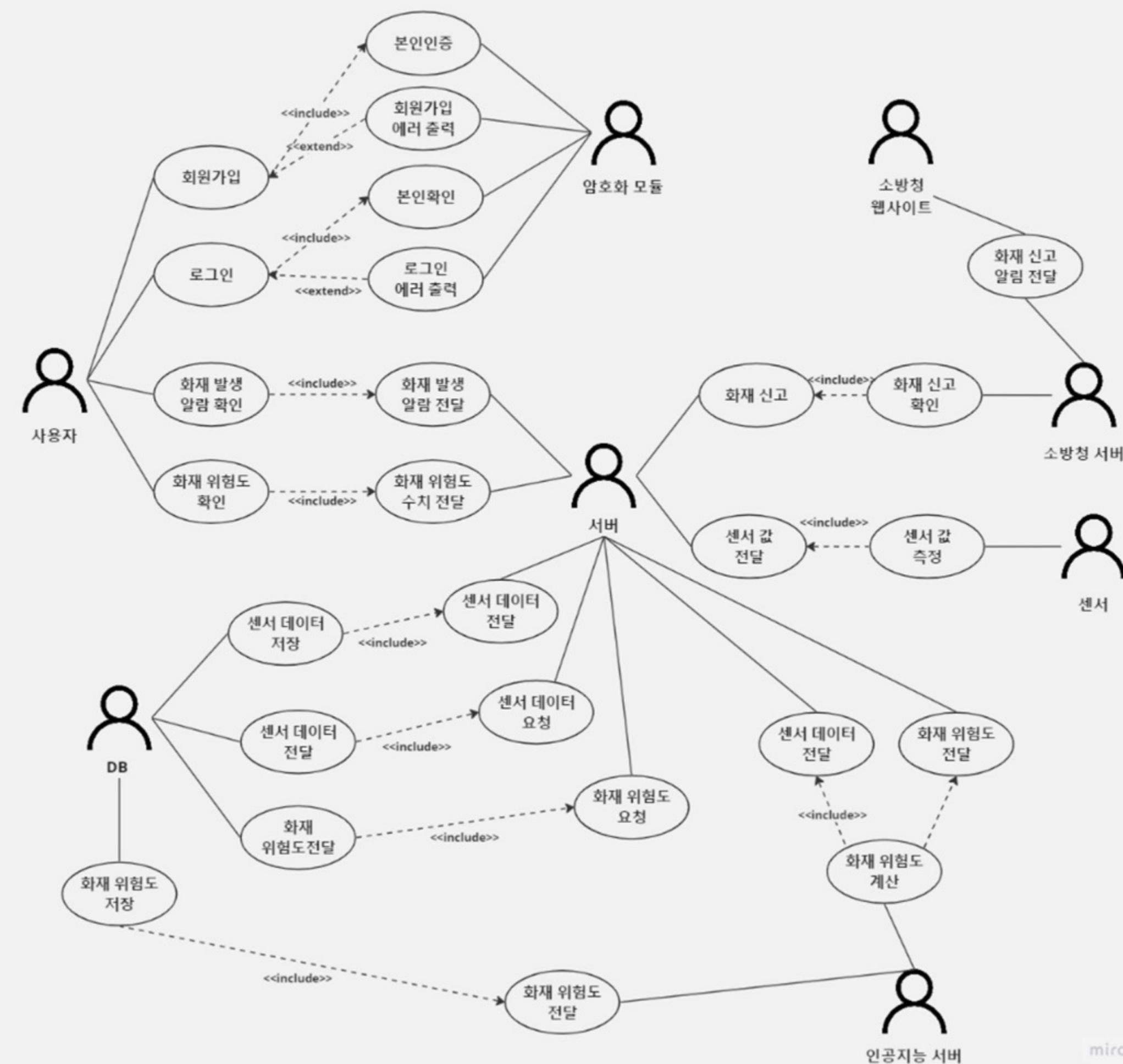
02

시스템 분석 설계 과정

● ● ● 02 | 시스템 분석 설계 과정_SWOT & CROSS SWOT

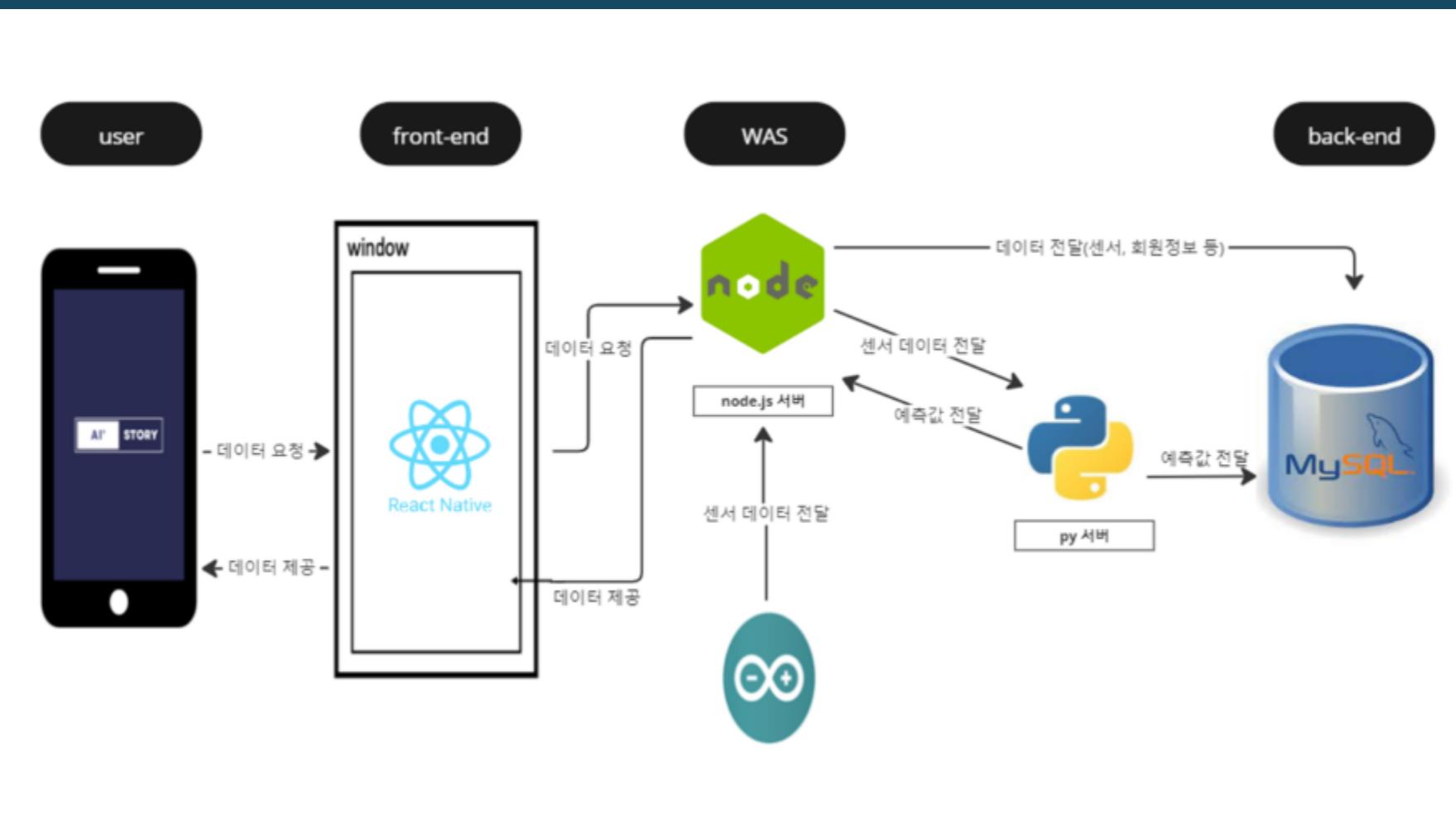


● ● ● 02 | 시스템 분석 설계 과정_유스케이스 다이어그램

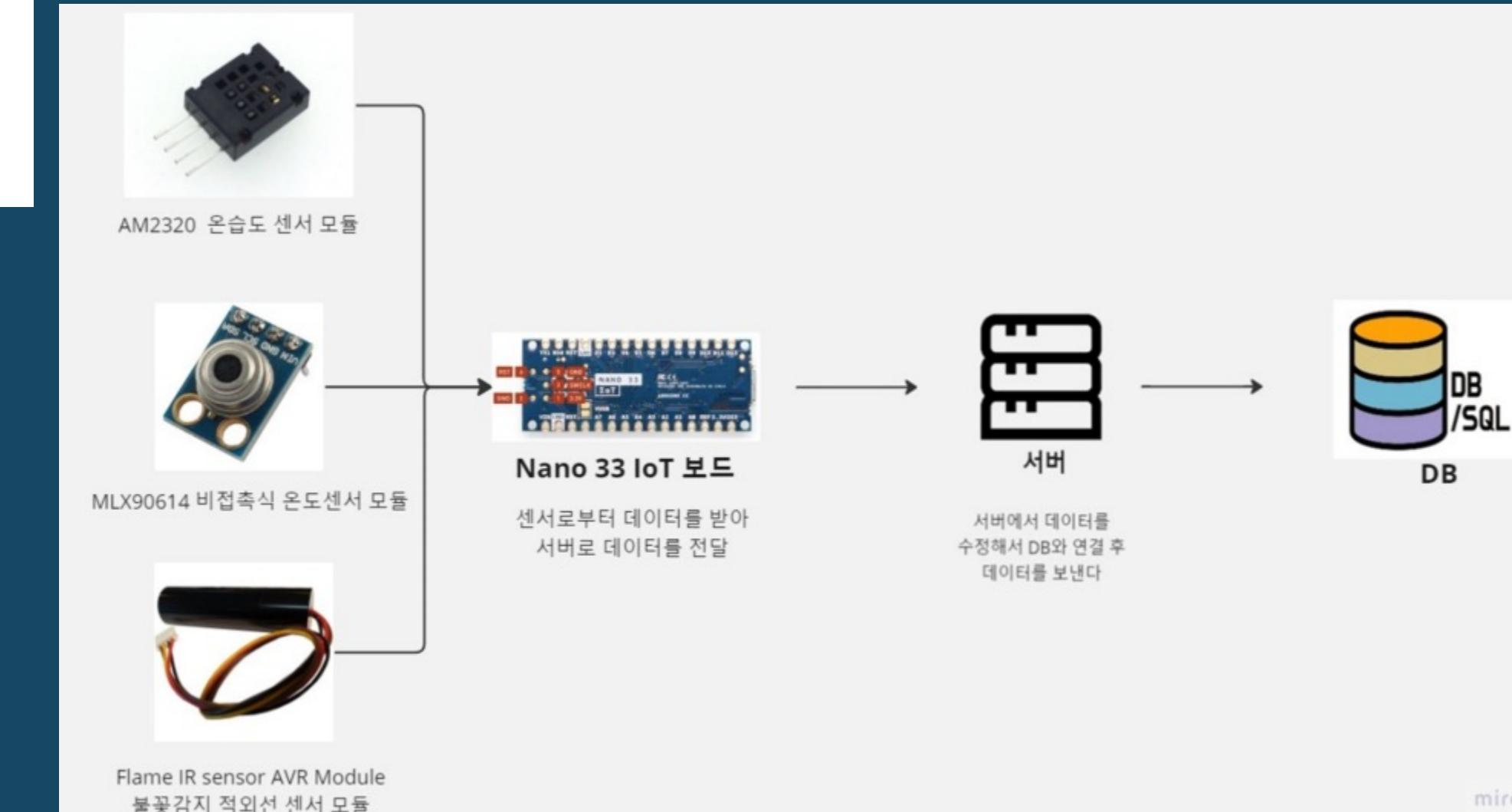


● ● ● 02 | 시스템 분석 설계 과정_아키텍처 설계

| 아키텍처 설계 - 서비스 구성도

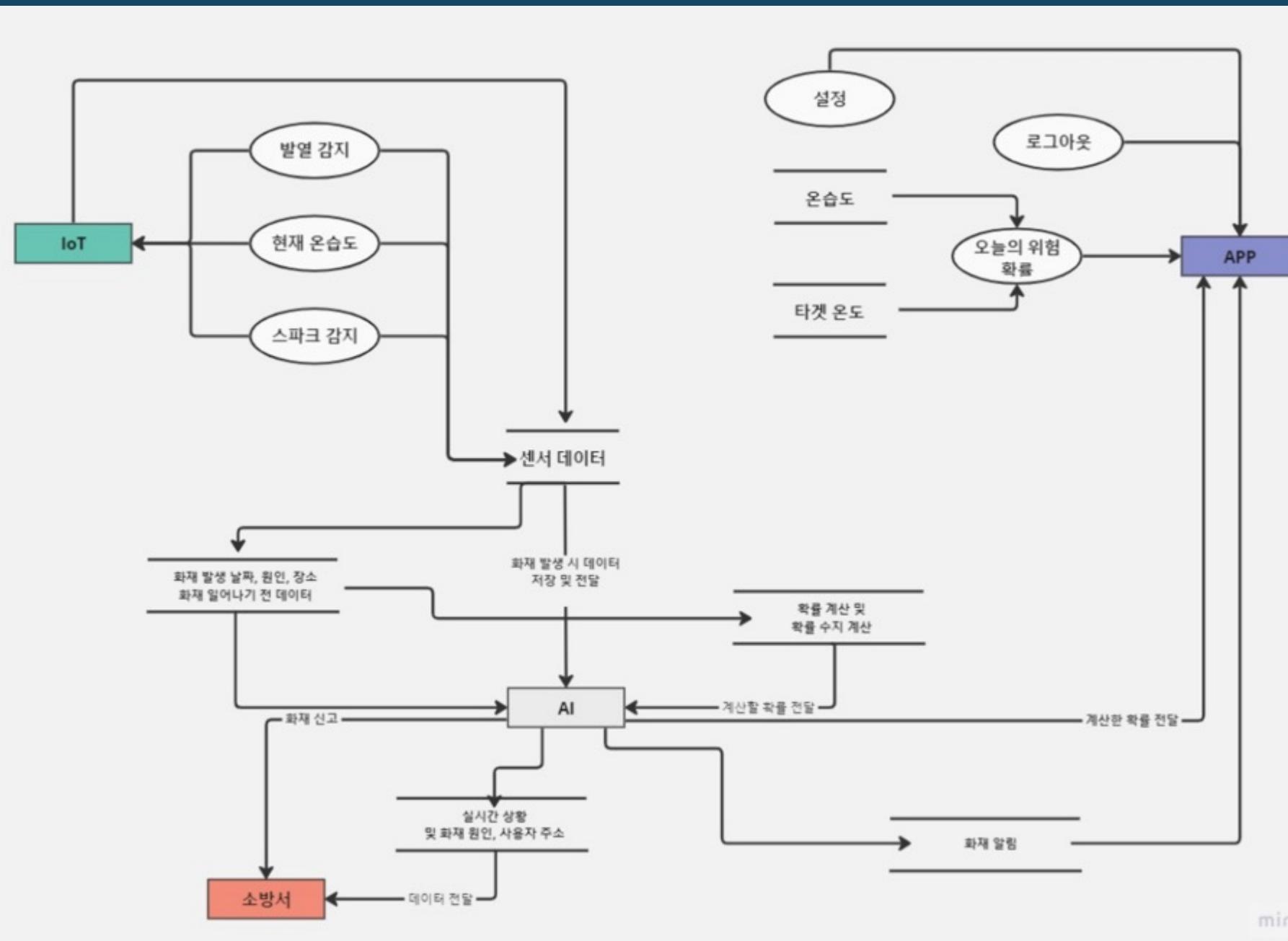


| 아키텍처 설계 - 센서 구성도



● ● ● 02 | 시스템 분석 설계 과정_아키텍처 설계

| 아키텍처 설계 - 데이터 흐름도



| 아키텍처 설계 - 인공지능 기능 흐름도



03

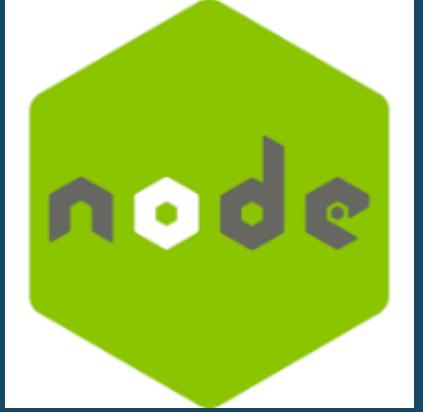
구현 및 기술

● ● ● 03 | 구현 및 기술_개발 환경 및 설명

구분	항목	적용내역	
S/W 개발환경	스마트폰 App 개발	React Native	application 프로그램 개발
		안드로이드OS(5.0.1)	스마트폰 운영체제
	인공지능 개발	PyTorch	PyTorch의 Net 클래스로 개발
	서버 & DB 개발	MySQL(5.1.73)	센서로부터 받아온 데이터를 저장, 관리하는 데이터베이스
		Node.js	어플리케이션 및 DB 연동
		Node.js(python server)	Python server를 사용하여 PyTorch를 사용한 인공지능과 DB 연동
H/W 구성장비	디바이스	스마트폰 (안드로이드)	사용자에게 서비스를 직접적으로 제공하는 End Device
		센서	비접촉식 적외선 온도센서, TB-I2C-H04, Flame IR Sensor AVR Module, AM2320 온습도 센서 모듈, Arduino Tiny Machine Learning Shield
	통신	Arduino Nano33 IoT	센서 데이터를 받아옴

● ● ● 03 | 구현 및 기술_개발 환경 및 설명

| 어플리케이션 개발



The mobile application interface includes:

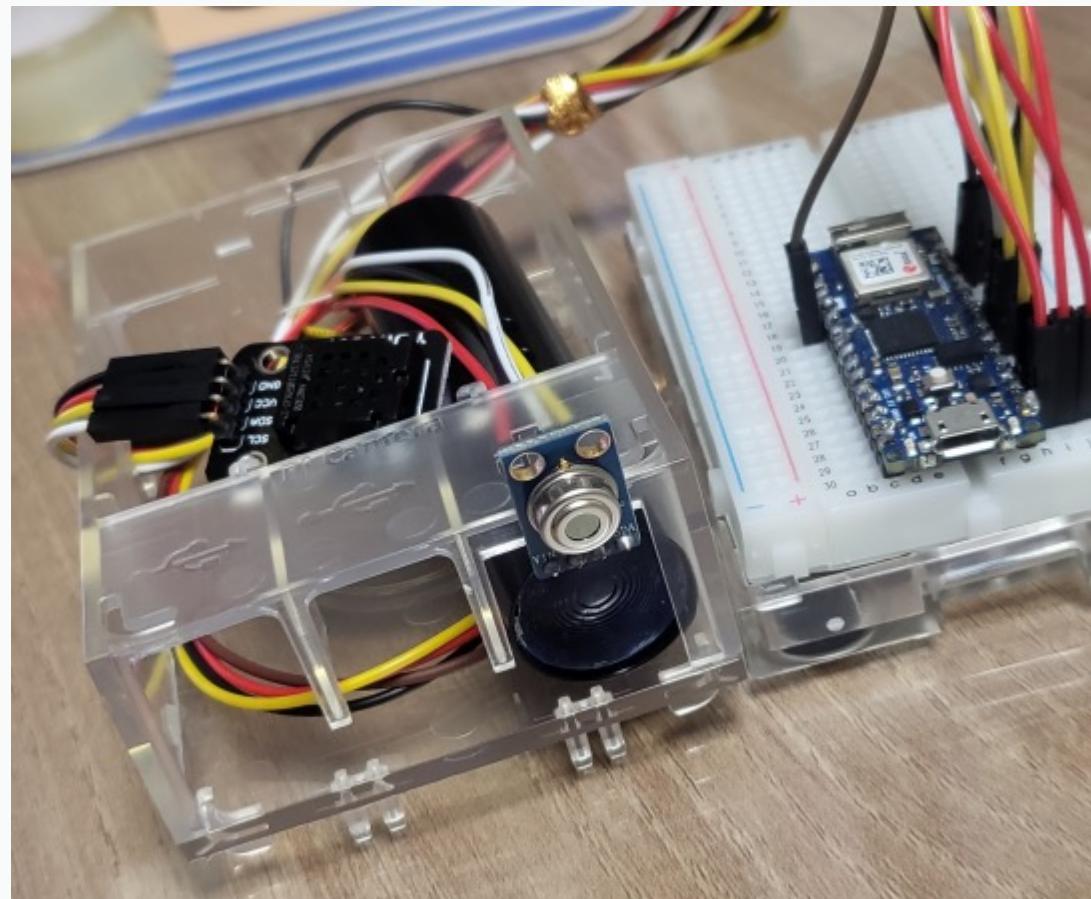
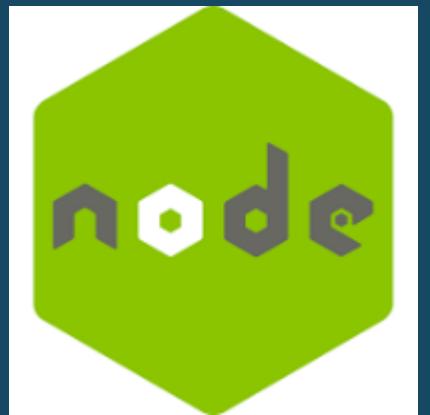
- 회원가입 (User Registration) screen with fields for 이름 (Name), 전화번호 (Phone Number), 비밀번호 (Password), 비밀번호 확인 (Password Confirmation), 주소 (Address), 비밀번호 확인 (Password Confirmation), 주소 (Address), and 센서번호 (Sensor Number). A "가입하기" (Sign Up) button is at the bottom.
- 신화면오 (New House) screen showing sensor numbers: 01012345678, 01011115555, a1234, a1234, and ss04.
- 로그인 (Login) screen with ID (01022223333) and Password (a1234). It shows a success message: "회원 가입 성공" (User registration successful) and "회원 가입이 성공적으로 완료되었습니다." (User registration completed successfully).
- 11월19일의 위험 확률 (Risk Probability for November 19) screen showing a 71% risk level with an orange box for "발열량 이상 발견" (Abnormal fever detected) and a note about persistent fever.
- 10월24일의 위험 확률 (Risk Probability for October 24) screen showing a 71% risk level with a red box for "화재 발생" (Fire occurrence) and a note about a fire at a residence.
- 09월19일의 위험 확률 (Risk Probability for September 19) screen showing a 32% risk level with a pie chart and a line graph.
- 오늘의 위험 확률 (Risk Probability for Today) screen showing a 32% risk level with a pie chart and a line graph.
- 온습도 버튼 클릭 시 (When the Humidity button is clicked) screen showing a pie chart with segments for 온습도 (Humidity), 발열량 (Fever), and 전체 (Total).
- 발열량 버튼 클릭 시 (When the Fever button is clicked) screen showing a pie chart with segments for 온습도 (Humidity), 발열량 (Fever), and 전체 (Total).
- 온습도 버튼 클릭 시 (When the Humidity button is clicked) screen showing a pie chart with segments for 온습도 (Humidity), 발열량 (Fever), and 전체 (Total).
- 화면 스크롤 시 앞의 3개 화면 모두 동일 (When the screen scrolls, the previous 3 screens are also the same) screen showing a large pink area with a pie chart.

-> 화재 감지는 되지 않았지만 발열량에 이상이 감지 되었을 때 뜨는 알림창

-> 화재 발생시 웹사이트에 화재 발생 주소지와 화재 알림을 띄우는 창

● ● ● 03 | 구현 및 기술_센서 구현

| 센서 구현



| 실제 구현된 센서

```
// 서버로 데이터 전송  
WiFiClient client;  
  
if (client.connect(serverAddress, serverPort)) {  
    client.println("POST /recieve HTTP/1.1");  
    client.println("Host: " + String(serverAddress));  
    client.println("Content-Type: application/json");  
    client.print("Content-Length: ");  
    client.println(data.length());  
    client.println();  
    client.println(data);
```

| 서버로 데이터를 전송하기 위한 아두이노 코드

Serial Monitor X Output

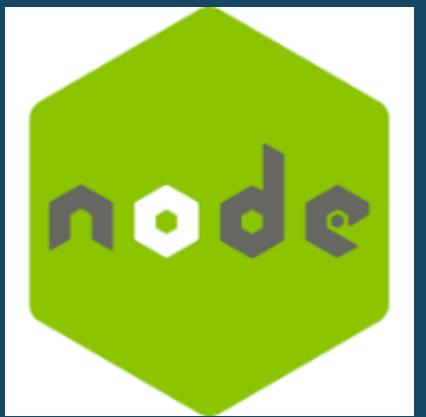
Message (Enter to send message to 'Arduino NANO 33 IoT' on 'COM3')

```
Access-Control-Allow-Origin: *  
Content-Type: application/json; charset=utf-8  
Content-Length: 52  
ETag: W/"34-dttHTFWApJdGry4AFK04RqlPdUI"  
Date: Mon, 18 Sep 2023 09:30:38 GMT  
Connection: keep-alive  
Keep-Alive: timeout=5
```

{"success":true,"message":"데이터 저장 성공"}Data sent successfully

| 서버에 데이터가 전송되면 뜨는 시리얼모니터 로그

| 인공지능 개발



과정 1) 과거 데이터 준비

```
# 데이터 로드 및 선처리  
data = pd.read_csv('/content/sample_data/newait2.csv')  
X = data[['season', 'flame_sensor_value', 'humidity', 'object_temp', 'ambient_temp']] -> 독립변수X  
X = pd.get_dummies(X, columns=['season'])  
y = data['fire']-> 종속변수Y
```

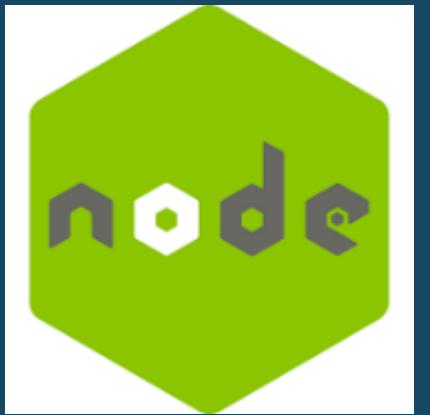
결측치 처리

```
X.fillna(X.mean(), inplace=True)
```

The screenshot shows two main parts of the 'Korea Fire Safety Big Data Platform'. On the left, the 'Data Market' section displays a search interface for datasets related to '도심형 화재발생 현황' (Urban fire occurrence status). It includes filters for '시군구', '화재 유형', and '화재 규모'. On the right, a detailed view of a specific dataset titled '도심형 화재발생 현황' is shown. This view includes sections for '파일 정보' (File Information), which lists the file name as '도심형 화재발생 현황' and the size as '556.30KB'; and '데이터 컬럼 정보' (Data Column Information), which lists various columns such as 'FIRE_TYPE_NM', 'TRCTORU_NM', 'BUILD_SRTRM', etc., along with their descriptions.

| 인공지능 학습용 데이터를 얻은 뒤
| 모델에 맞춰 전처리

| 인공지능 개발



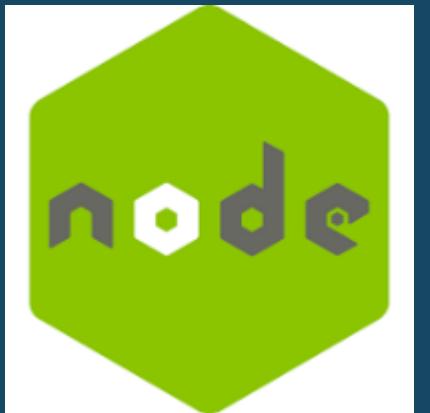
과정 2) 모델 구조 생성

```
# 신경망 모델 정의
class Net(nn.Module):
    def __init__(self, input_size):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_size, 128)
        self.dropout1 = nn.Dropout(0.3)
        self.fc2 = nn.Linear(128, 64)
        self.dropout2 = nn.Dropout(0.3)
        self.fc3 = nn.Linear(64, 32)
        self.dropout3 = nn.Dropout(0.3)
        self.fc4 = nn.Linear(32, 1)
        self.tanh = nn.Tanh()

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.dropout1(x)
        x = torch.relu(self.fc2(x))
        x = self.dropout2(x)
        x = torch.sigmoid(self.fc3(x))
        x = self.dropout3(x)
        x = self.fc4(x)
        x = self.tanh(x)
        return x
```

- > 인공지능의 신경망 모델 (PyTorch_Net() 사용)
- > 출력층 활성화 함수 tanh 사용

| 인공지능 개발



과정 3) 모델 학습 및 평가

```
# 모델 학습
input_size = X_train.shape[1]
model = Net(input_size)
criterion = FocalLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, patience=10, verbose=True)

num_epochs = 1000
early_stopping_counter = 0
best_loss = float('inf')

for epoch in range(num_epochs):
    inputs = torch.tensor(X_train.values, dtype=torch.float32)
    labels = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)

    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
```

-> 신경망 모델, 손실 함수, 최적화 프로그램 및 학습률 스케줄러 초기화

```
# 조기 종료
if loss < best_loss:
    best_loss = loss
    early_stopping_counter = 0
else:
    early_stopping_counter += 1

if early_stopping_counter >= 20:
    print(f"Early stopping at epoch {epoch}")
    break
```

-> 조기종료 함수 정의

| 인공지능 개발



과정 3) 모델 학습 및 평가

```
# 모델 평가
with torch.no_grad():
    mse = mean_squared_error(y_test, predicted_prob)
    r2 = r2_score(y_test, predicted_prob)

    rmse = np.sqrt(mse)

    print("Mean Squared Error (Probability):", mse)
    print("R-squared (R2) Score (Probability):", r2)
    print("Root Mean Squared Error (RMSE) (Probability):", rmse)

results_prob_df = pd.DataFrame({'Actual': y_test.values, 'Predicted_Prob': predicted_prob.flatten()})

# risk levels
def categorize_risk(probability):
    if probability <= 0.3:
        return 'Low Risk'
    elif probability <= 0.65:
        return 'Medium Risk'
    elif probability <= 0.7:
        return 'High Risk'
    else:
        return 'Very High Risk'

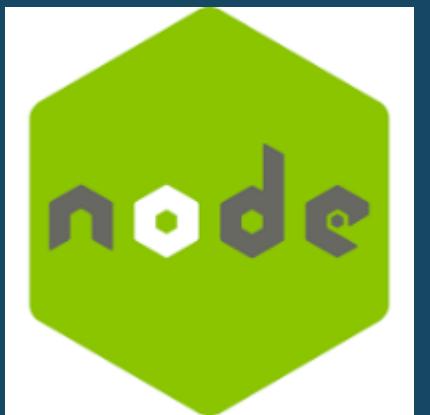
results_prob_df['Risk_Level'] = results_prob_df['Predicted_Prob'].apply(categorize_risk)

print("\nProbability Predictions:")
print(results_prob_df)

results_binary_df = pd.DataFrame({'Actual': y_test.values, 'Predicted_Binary': predicted_binary.flatten()})
# print("\nBinary Predictions:")
# print(results_binary_df)
```

-> 평균 제곱 오차, R제곱 및 평균 제곱근 오차를 사용하여 test 셋에서 훈련된 모델을 평가

| 인공지능 개발



과정 4) 모델 예측(이용)

```
# 새로운 데이터에 대한 예측
with torch.no_grad():
    new_inputs = torch.tensor(new_X.values, dtype=torch.float32)
    new_predicted_logits = model(new_inputs)
    new_predicted_prob = torch.sigmoid(new_predicted_logits).numpy()

    threshold = 0.5
    new_predicted_binary = (new_predicted_prob > threshold).astype(int)

# 예측 결과 출력
new_results_df = pd.DataFrame({'Predicted_Prob': new_predicted_prob.flatten()})
# new_results_df['Risk_Level'] = new_results_df['Predicted_Prob'].apply(categorize_risk)
print("Predictions for new data:")
print(new_results_df)

import torch.jit

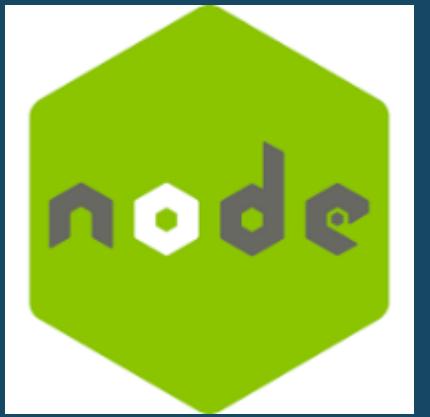
# 모델을 torch.jit.ScriptModule으로 변환하여 저장
scripted_model = torch.jit.script(model)
torch.jit.save(scripted_model, 'model_5.pth')

torch.save(model.state_dict(), 'model_weights_5.pth')
```

-> 훈련된 모델을 사용하여
새 데이터에 대한 확률과 이진 결과를 예측

-> PyTorch 모델을 스크립트 모듈로 변환하여 저장

| 인공지능 개발



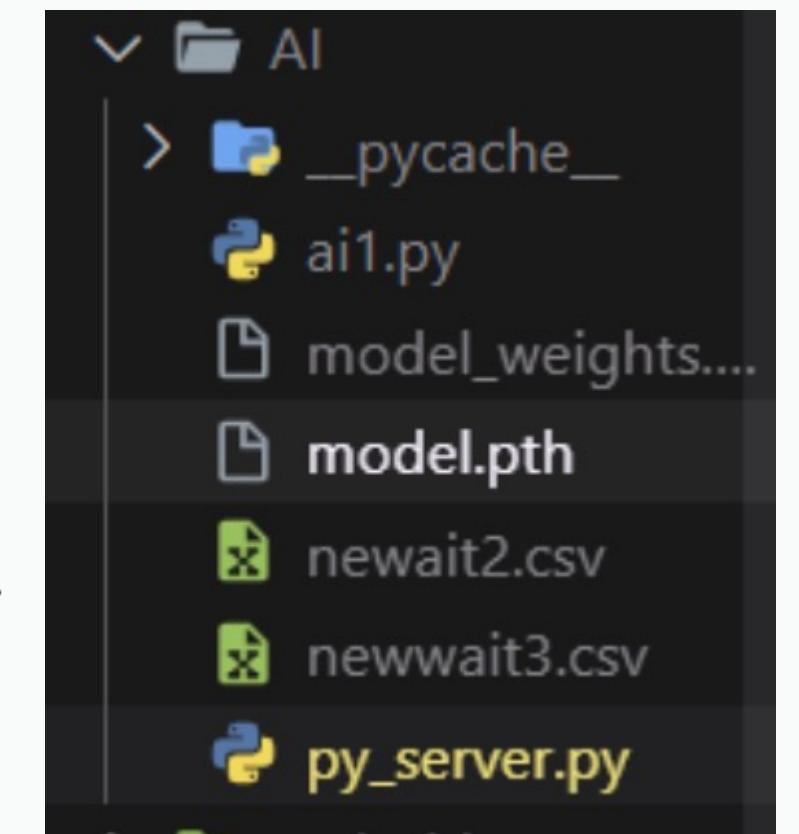
과정 5) 모델 및 가중치 추출

```
209  
210  
211 # import torch.jit  
212  
213 torch.save(model.state_dict(), 'model17.pth')  
214 # print.loaded_model)  
215 # torch.save(model.state_dict(), 'model3.pth')  
216 # torch.save(model.state_dict(), 'model10.pth')  
217 # print.loaded_model)  
218  
219 loaded_model = Net(input_size)  
220  
221 loaded_model.load_state_dict(torch.load('model17.pth'))  
222  
223 print(loaded_model)  
224 -> 인공지능의 모델과 가중치 추출  
225
```

```
5 # print()  
6 import joblib  
7 scaler_path = 'scaler5.pkl'  
8 joblib.dump(scaler, scaler_path)  
9  
-> 데이터 전처리를 위한 scaler 추출
```

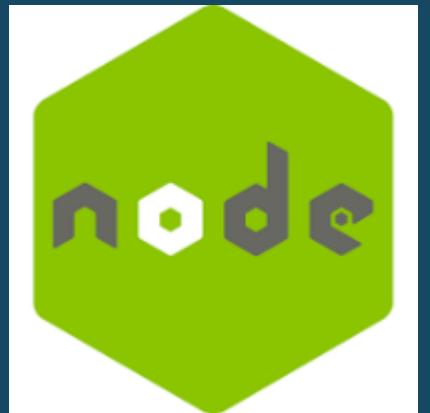
```
Net(  
    (fc1): Linear(in_features=8, out_features=128, bias=True)  
    (dropout1): Dropout(p=0.3, inplace=False)  
    (fc2): Linear(in_features=128, out_features=64, bias=True)  
    (dropout2): Dropout(p=0.3, inplace=False)  
    (fc3): Linear(in_features=64, out_features=32, bias=True)  
    (dropout3): Dropout(p=0.3, inplace=False)  
    (fc4): Linear(in_features=32, out_features=1, bias=True)  
    (tanh): Tanh()  
)
```

-> 추출한 모델과 가중치의 형태



어플리케이션에 model과
가중치 추출한 것 저장->

| 인공지능 개발



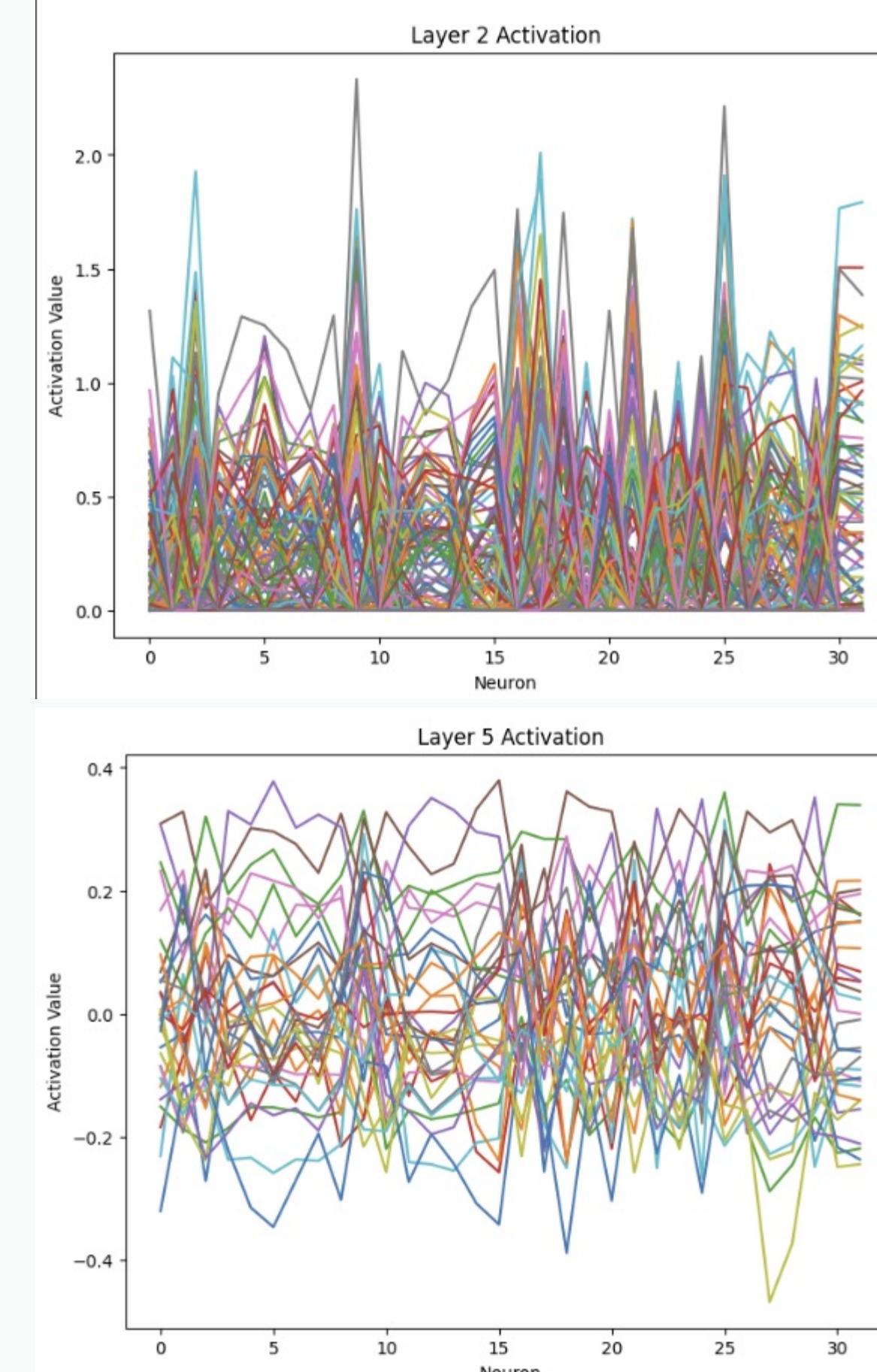
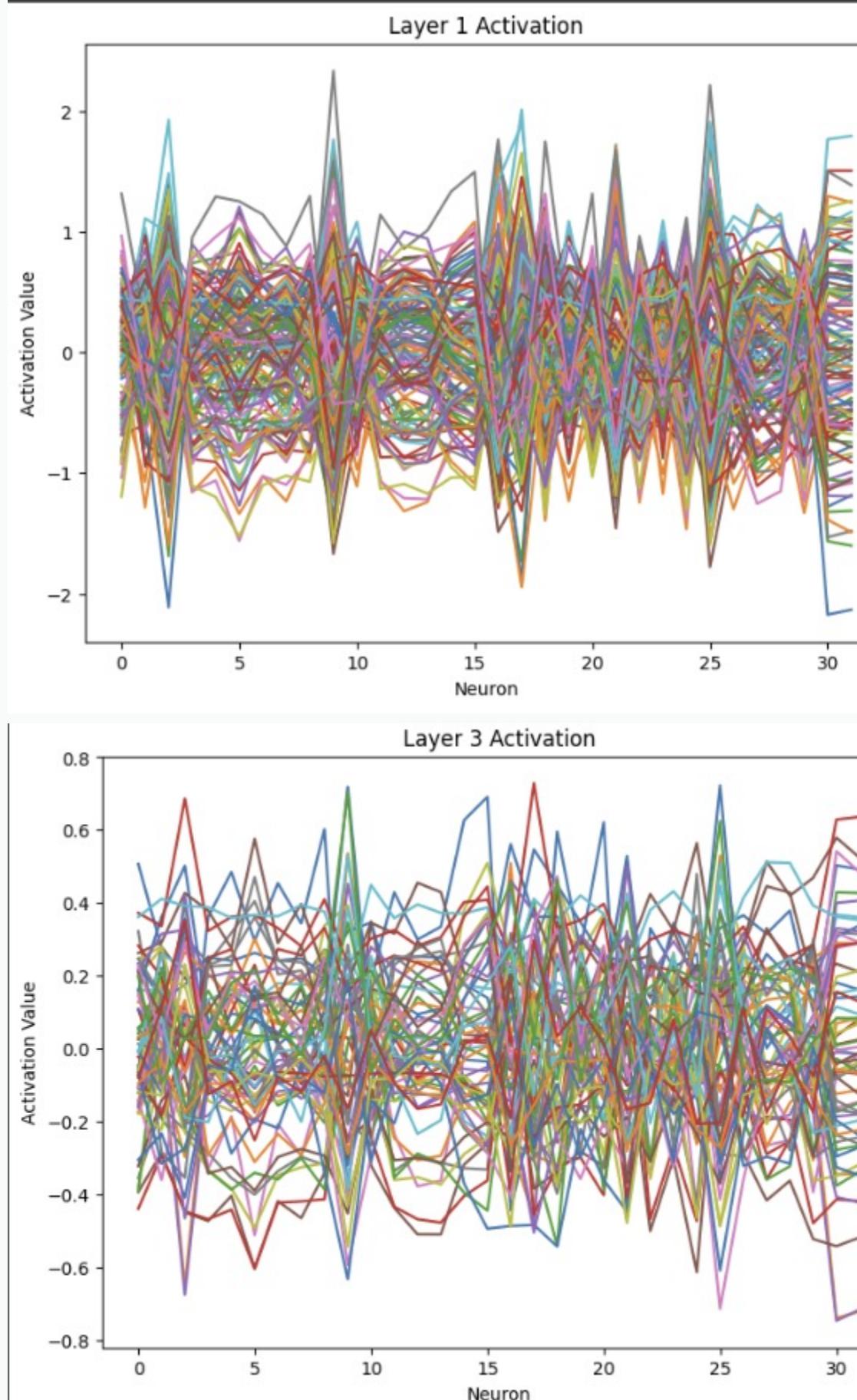
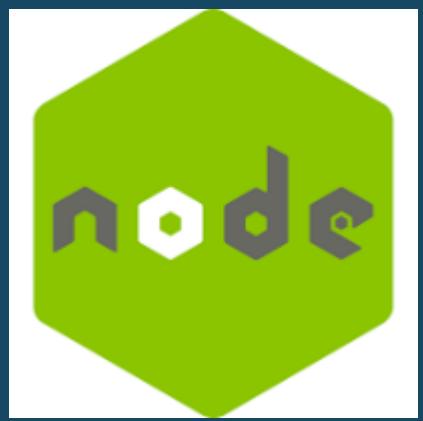
| 실제 예측값이 어플을 통해 사용자에게 제공되는 부분

```
python 서버 응답: { predicted_prob: 0.7207356691360474
python 서버 응답: { predicted_prob: 0.7207356691360474
python 서버 응답: { predicted_prob: 0.7207356691360474
S C:\Users\USER\aistory_2> node server.js
server is running on http://localhost:3000
connected to MySQL database []
python 서버 응답: { predicted_prob: 0.7207356691360474
.
S C:\Users\USER\aistory_2> node server.js
server is running on http://localhost:3000
connected to MySQL database
python 서버 응답: { predicted_prob: 0.27227678894996643
}
S C:\Users\USER\aistory_2>
노드에서 전달받은 데이터: {'sensor_data': [{'season': 'Winter', 'flame_sensor_value': 1, 'humidity': 57.4, 'object_temp': 50, 'ambient_temp': 109.99}]}
sensor_data {'season': 'Winter', 'flame_sensor_value': 1, 'humidity': 57.4, 'object_temp': 50, 'ambient_temp': 109.99}
tensor([[ 0.0000,  1.0000,  0.0000,  0.0000,  1.0000,  57.4000,  50.0000,
         109.9900]])
인공지능 모델 예측 결과: 0.7207356691360474
데이터 저장 성공: 72
192.168.35.45 - - [19/oct/2023 23:30:36] "POST /predict HTTP/1.1" 200 -
노드에서 전달받은 데이터: {'sensor_data': [{'season': 'Autumn', 'flame_sensor_value': 0, 'humidity': 57.4, 'object_temp': 30, 'ambient_temp': 30}]}
sensor_data {'season': 'Autumn', 'flame_sensor_value': 0, 'humidity': 57.4, 'object_temp': 30, 'ambient_temp': 30}
tensor([[ 1.0000,  0.0000,  0.0000,  0.0000,  0.0000,  57.4000,  30.0000,  30.0000]])
인공지능 모델 예측 결과: 0.27227678894996643
데이터 저장 성공: 27
192.168.35.45 - - [19/oct/2023 23:31:30] "POST /predict HTTP/1.1" 200 -
```

| 어플리케이션 서버와 인공지능 서버 간에 센서 데이터
와 인공지능의 예측값을 주고 받을 때의 로그

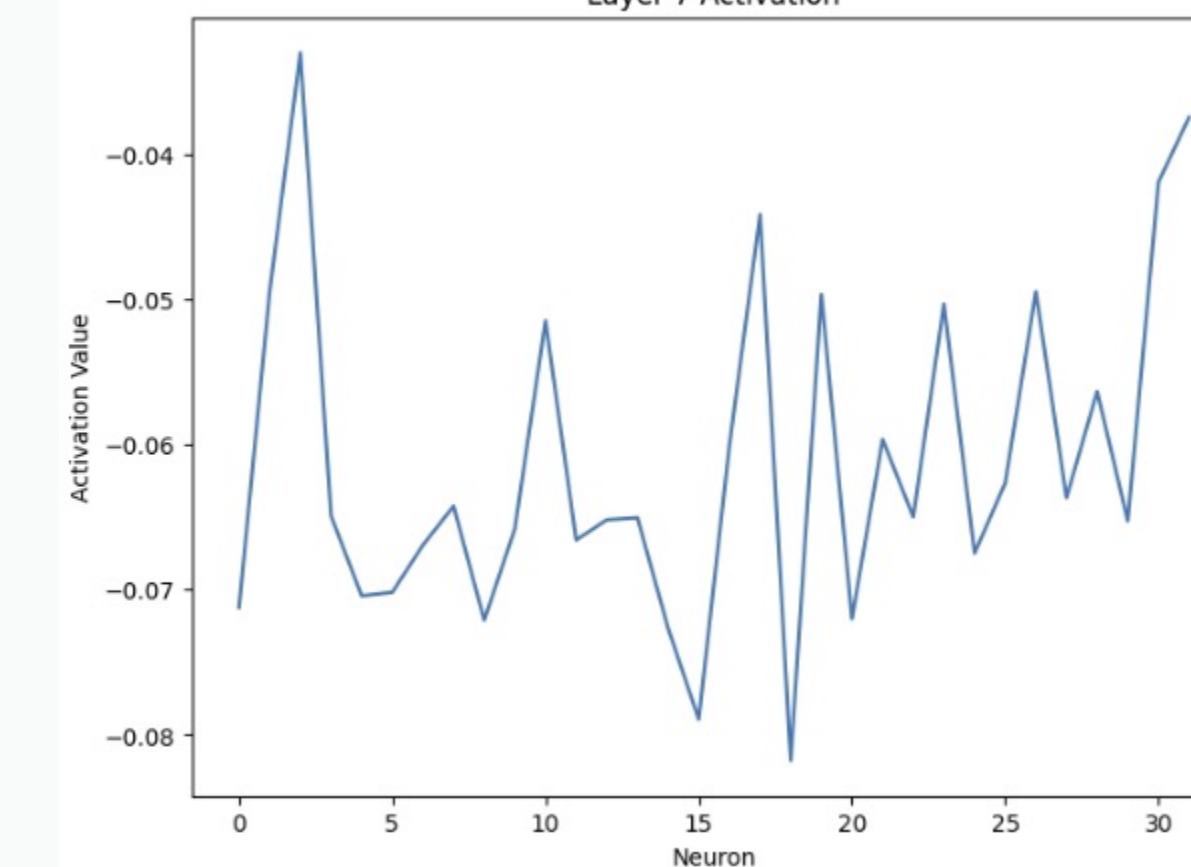
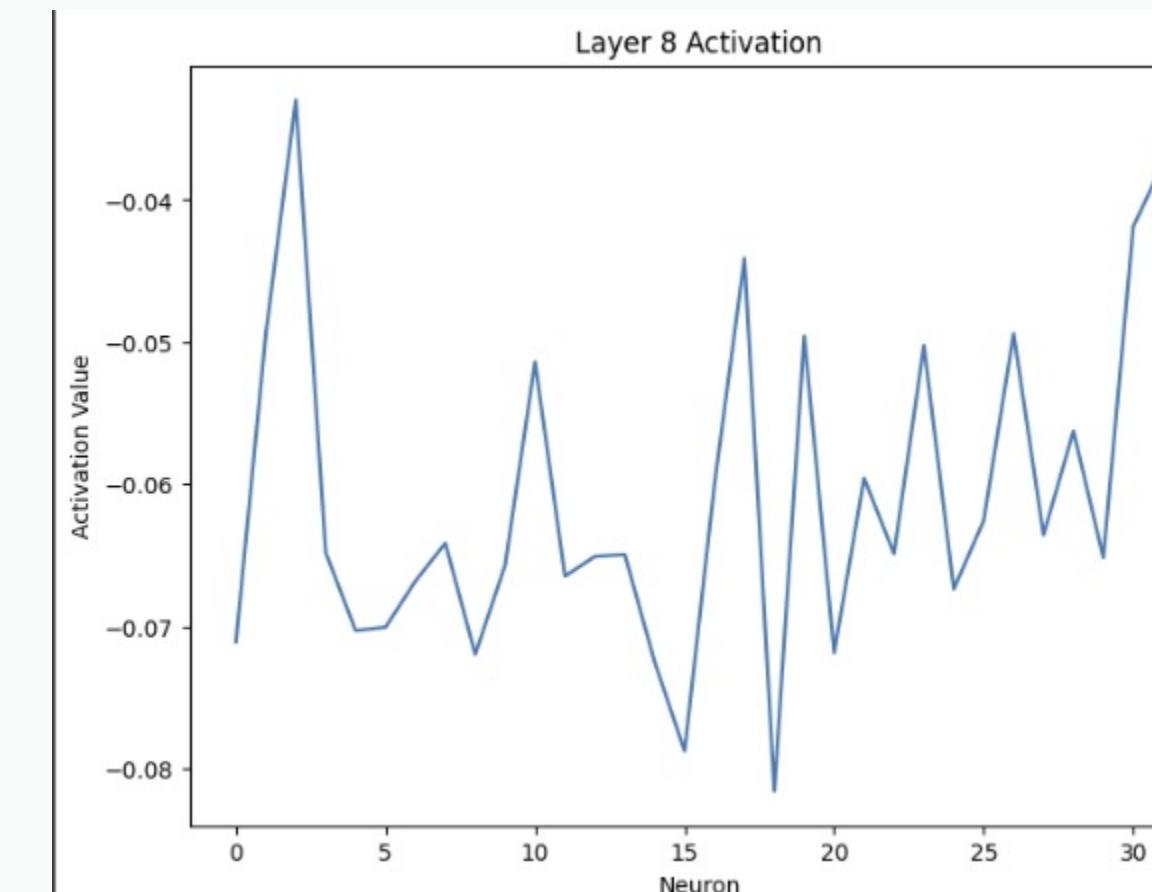
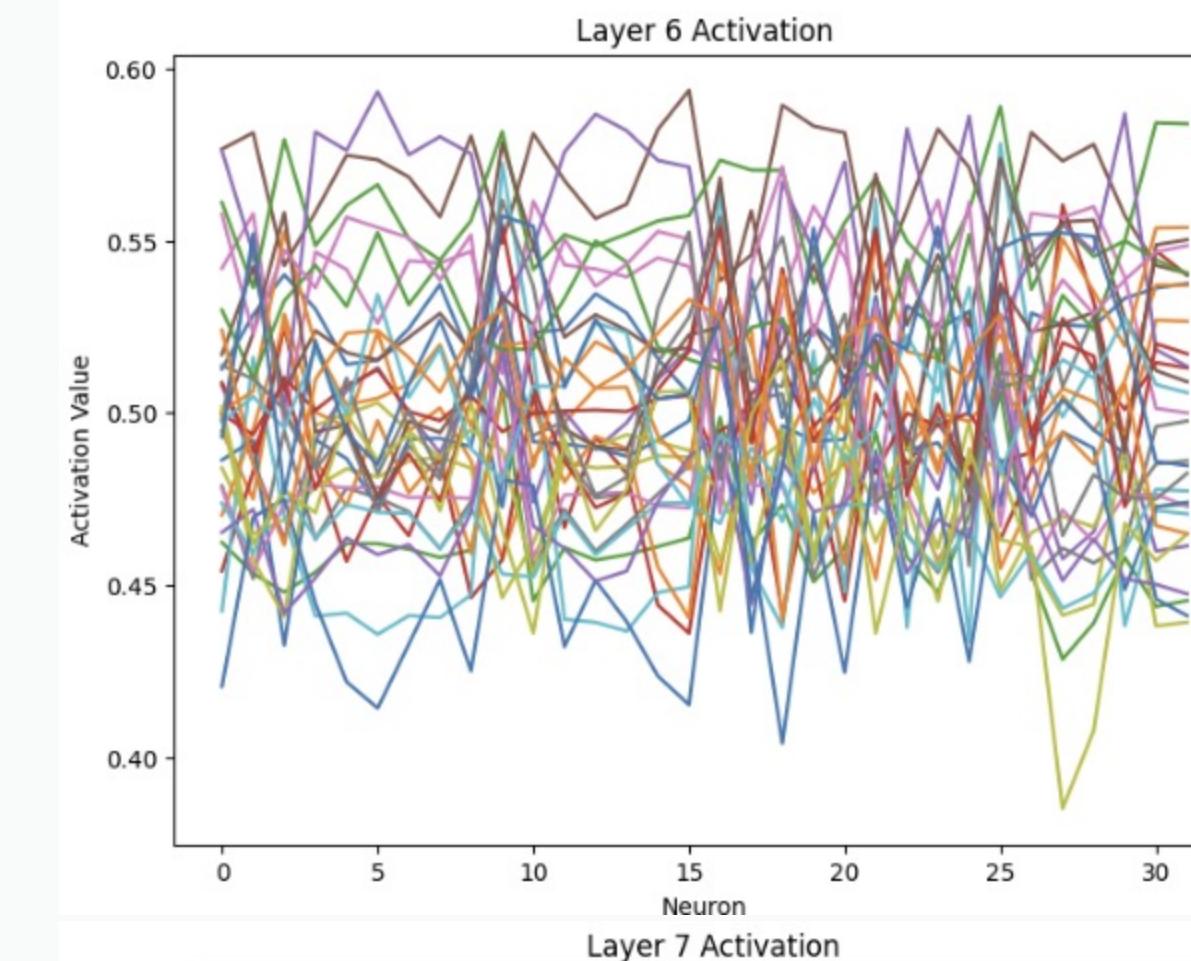
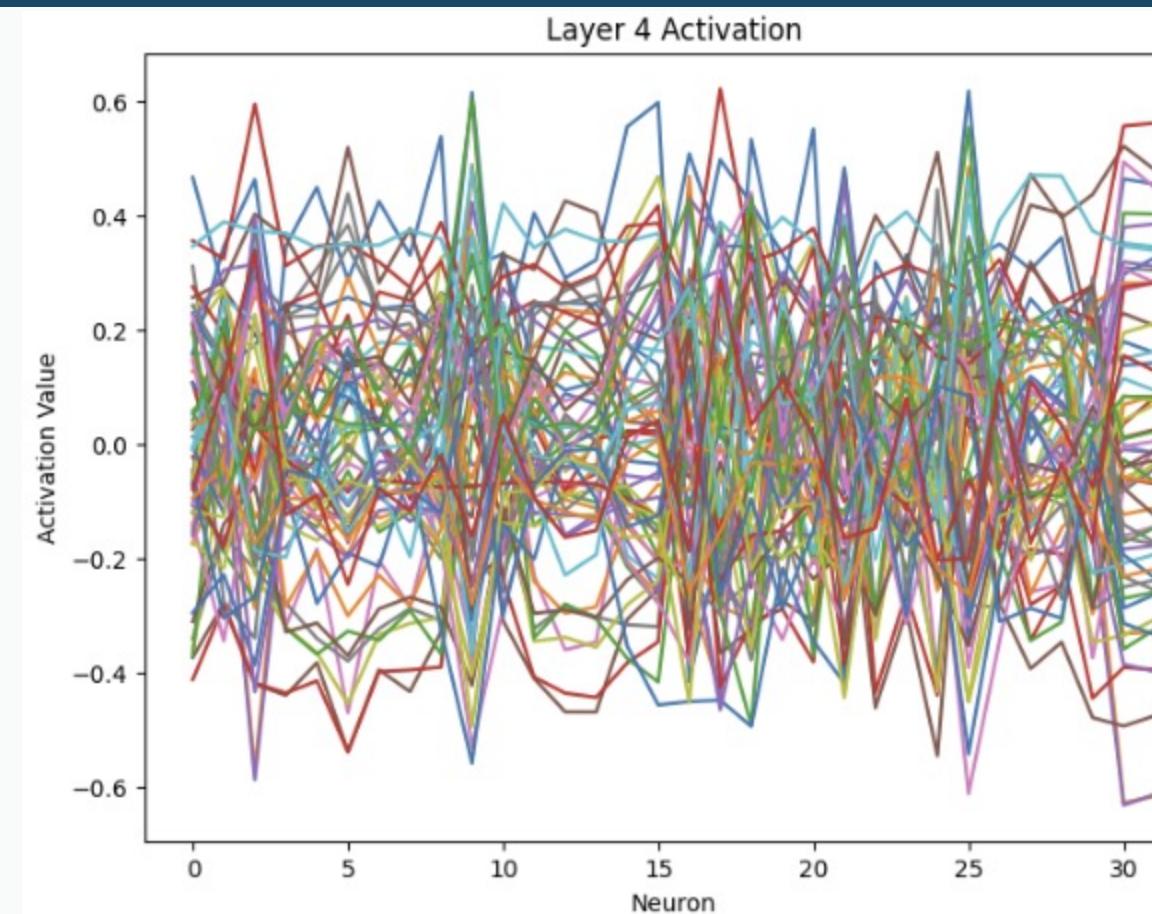
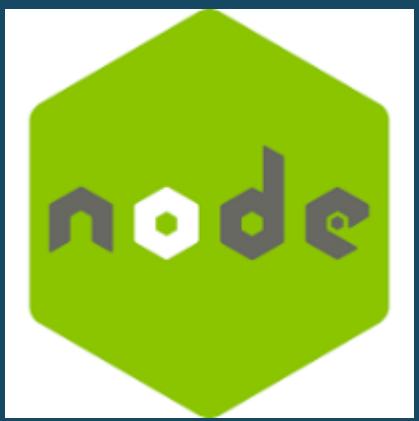
● ● ● 03 | 구현 및 기술_인공지능 구현 시작화

| 인공지능 개발

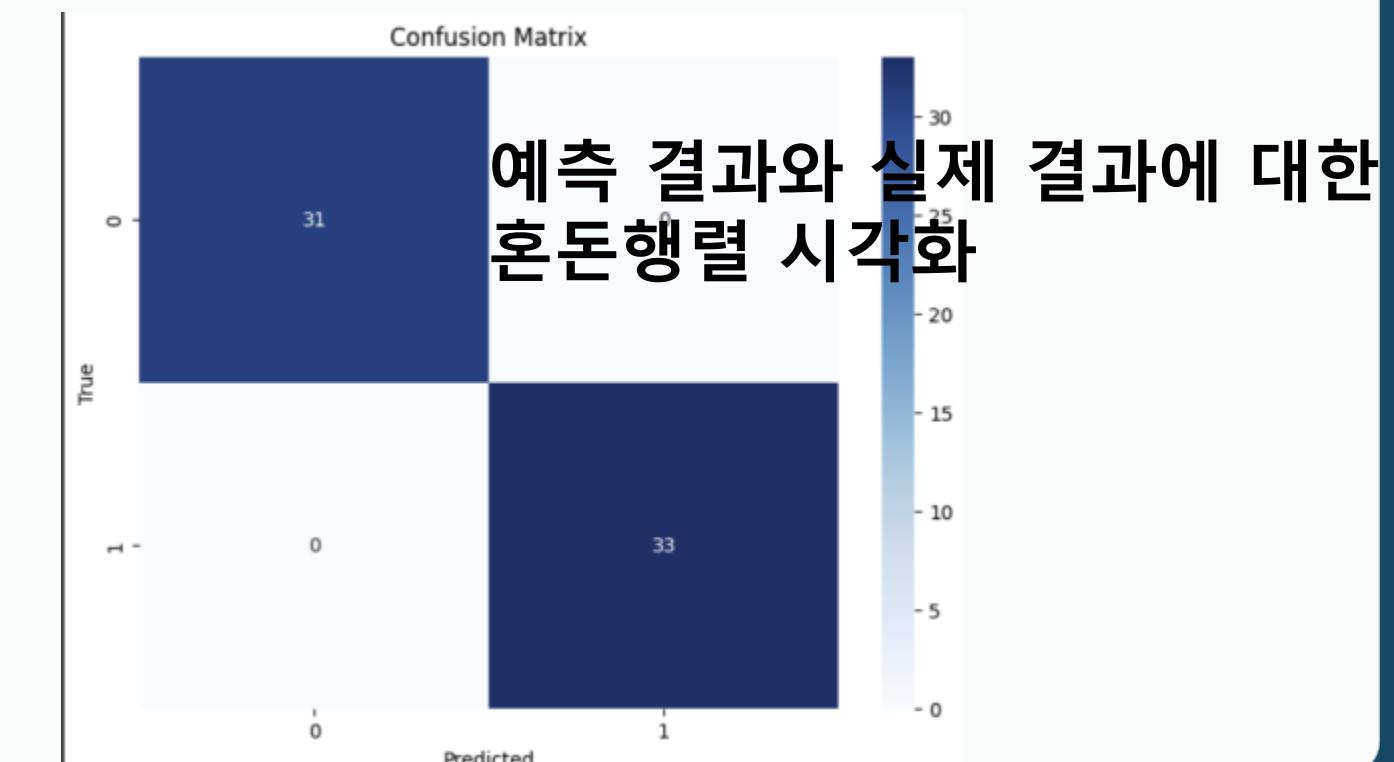
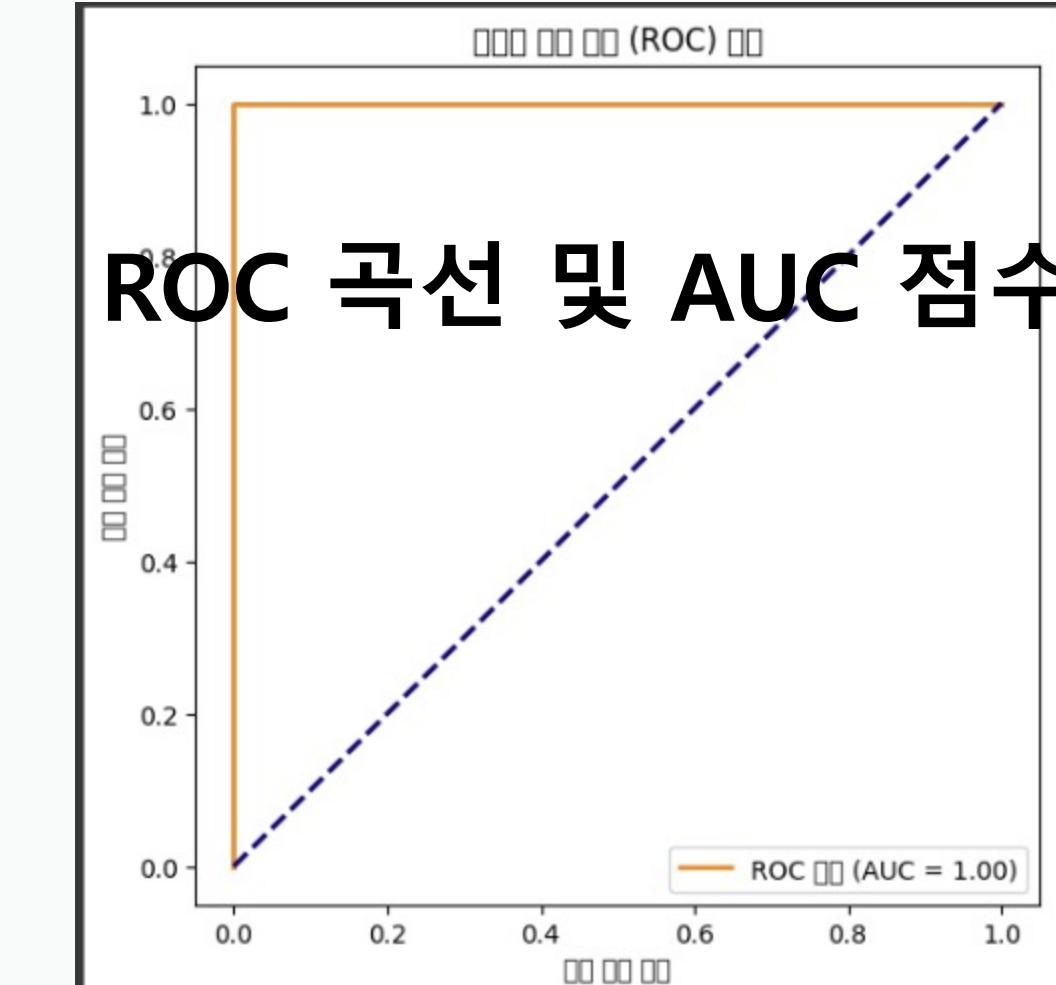
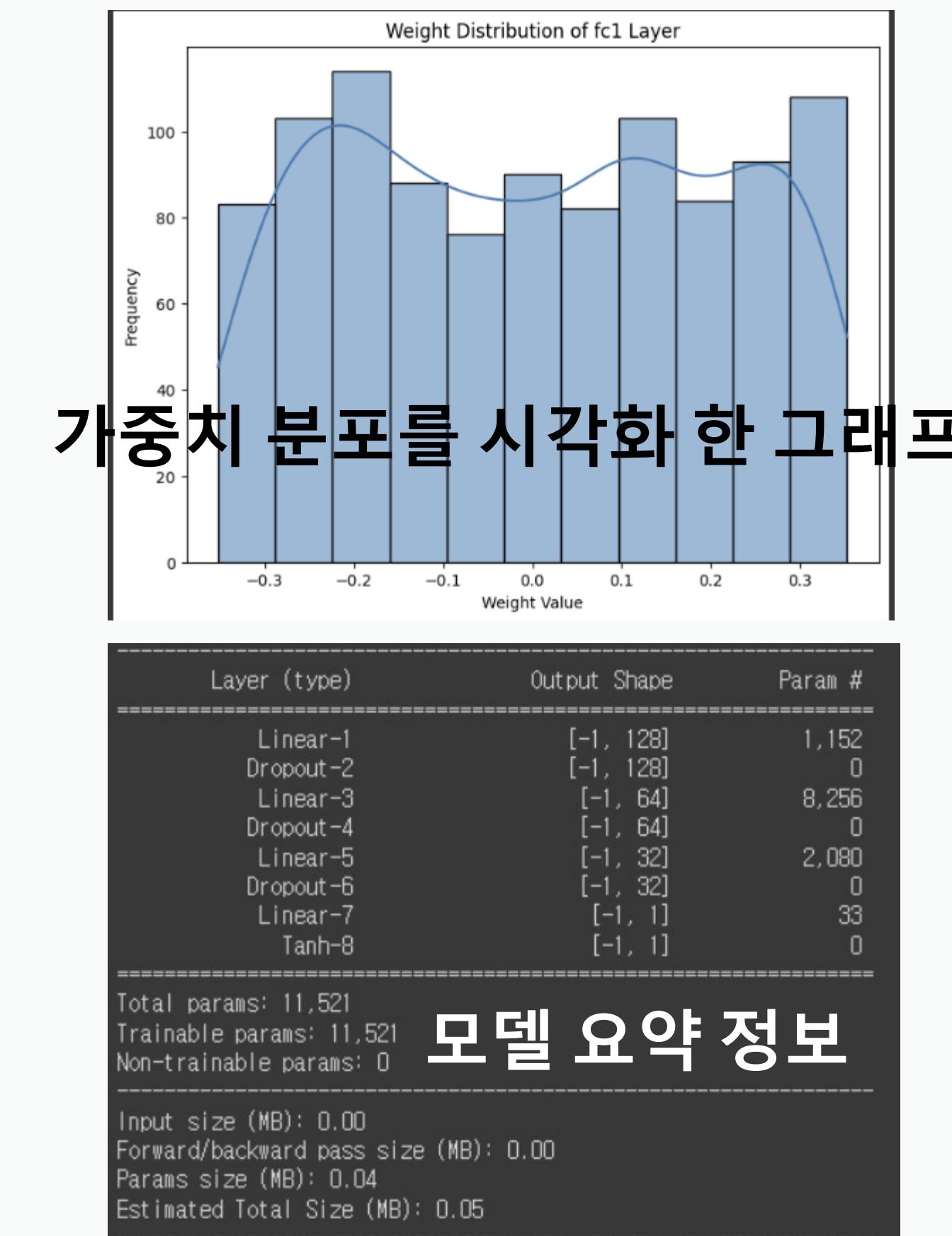
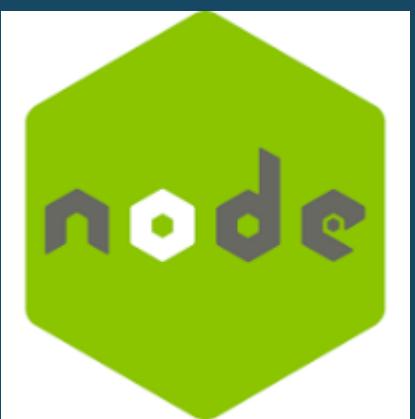


● ● ● 03 | 구현 및 기술_인공지능 구현 시작화

| 인공지능 개발



| 인공지능 개발



04

기대효과

사회적 기대효과

- 전선의 접합부 온도와 스파크 등을 독립변수로 하여 화재 확률을 계산하는 인공지능의 경우 전기적 화재에 대해 전보다 명확한 데이터를 수집할 수 있어 화재의 원인이 전기적 화재임을 확실하게 알 수 있음.
-> 전기적 화재의 미확인 단락만 크게 줄여도 전기화재 빈도를 크게 낮출 수 있음.
- 센서로부터 추출한 전기적 화재에 대한 빅데이터를 공기업 또는 사기업에 제공하거나 판매할 수 있다.
-> 판매 수익으로 AIoT의 기능을 업그레이드하거나 추가로 물량을 확보하여 다시 사회적 소외계층에게 보급 할 수 있는 순환 구조가 형성되는 것을 기대할 수 있음.
- 화재시 합판, 비닐, 스티로폼 등이 타면서 발생하는 유해 물질은 산불 연기보다 훨씬 유독한 연기이고, 근처 하천이나 토양을 오염시켜 수질 오염을 일으킨다.
-> 오염물질이 어류와 저서생물 체내에 축적되어 음식을 통해 사람 몸으로 들어올 수 있다.
- 사회적 소외계층의 안전을 위한 화재 예측 감지 AIoT 플랫폼을 사용한다면 신고 시간이 단축되어 피해 면적이 줄어들 것이고, 이는 환경 오염 및 인체에도 피해를 줄일 수 있을 것이다.

사회적 기대효과 근거 자료

한편, 산불을 진화하는 과정에서 소방관뿐만 아니라 지역 주민이 자발적으로 진화에 참여하는데, 마스크를 착용하더라도 호흡을 통해 오염물질에 많이 노출될 수 있다. 산불 연기를 직간접적으로 흡입하는 것은 담배 연기를 마시는 것과 비슷한 효과가 있다. 산불이 아니라도 화재가 발생하면 유해물질에 노출될 가능성이 크다. 특히, 폐목재, 타이어, 비닐, 플라스틱 제품을 농업 부산물과 함께 태울 때는 조심해야 한다.

시골에서 온갖 쓰레기를 함께 소각하는 경우를 종종 볼 수 있는데, 이때 산불 연기보다 훨씬 유독한 연기가 나온다. 불법 소각 여부를 떠나서 주민 건강에 직접적인 영향을 미칠 수 있으므로 이에 대한 인식 개선이 필요하다.

특히, 불법 소각으로 배출된 오염물질은 주변 농토와 농작물에 침적되고, 먹이 사슬을 통해 다시 인체로 유입되는 사실을 명심해야 한다.

이처럼 시골이나 산악지역에서는 산불이나 농업 연소에 의한 오염물질 배출이 상당하다. 한편, 도심에서는 자동차뿐만 아니라 식당가 직화구이와 같은 생물성 연소를 통해 배출되는 오염물질도 무시할 수 없다. 정리하자면, 무엇이라도 태우면 오염물질이 발생하고, 결국 공기와 음식을 통해서 우리 몸으로 다시 들어온다.

전기안전공사에 따르면 미확인단락으로 분류된 화재 가운데 정밀조사를 실시한 결과 전기 외에 다른 요인으로 발생한 화재도 발견되고 있다. 전선에 녹은 흔적이 발생했지만, 불로 인해 열이 높아져서 용융흔이 생긴 경우가 적지 않다. 전기적 요인으로 일어난 화재가 아님에도 전기화재로 분류되고 있다는 얘기다.

전기안전공사는 이 같은 미확인단락 분류가 전기화재 점유율 감소를 막는 주요 원인 중 하나로 지목하고 있다. 전기화재를 원인별로 분석한 결과 미확인단락에 의한 화재가 절연열화에 의한 단락 다음으로 빈번하게 발생했다.

이와 관련 전기안전공사 산하 전기안전연구원(원장 최종수)은 보다 정확한 화재조사를 위해 힘쓰고 있다.

미확인단락은 명확한 원인을 밝히지 못했지만 '전기화재에 의한 발화로 추정되는 화재'를 의미한다. 이 미확인단락은 가장 큰 전기화재 원인인 절연열화에 의한 단락 다음으로 큰 비중을 차지하고 있다. 미확인단락만 크게 줄여도 전기화재 빈도를 크게 낮출 수 있다는 것.

2010년 전체 전기화재 건수는 9442건 정도였으며, 그해 발생한 미확인단락 화재는 2013건으로 21.3%에 달했다. 이듬해인 2011년 9351건의 전기화재 중에 2245건의 원인을 알 수 없는 전기화재가 발생했다. 전체의 24%에 달하는 수치다. 2012년과 2013년도 9225건과 8889건의 전기화재 가운데, 미확인단락이 각각 2396건(26%)과 2207건(24.8%) 발생했다. 2014년은 전기화재 8287건 가운데 25%인 2074건이 미확인단락으로 분류됐다.

전기화재나 전기화재 점유율은 지속적으로 줄고 있지만 미확인단락에 의한 화재 비중은 20% 초반대를 꾸준히 유지하고 있다. 무엇보다 원인을 정확히 알 수 없는 화재 요인이 20% 이상이나 된다는 점도 문제다. 이 같은 미확인단락을 통계에 집어넣은 분야도 전기화재가 유일하다.

경제적 기대효과

- 소방청과 사용자 사이에 더 긴밀하고 정확한 정보를 주고 받을 수 있는 **연결망을 구축하는 역할 가능**
- 화재는 수많은 **재산 피해와 인명 피해**를 동반함.
-> 화재로 인한 피해 면적을 줄인다면 재산 피해나 인명 피해도 자연스레 줄어들게 될 것이다.
- 실제로 소방청은 소방공무원의 **신속한 초기 진화 등으로 경감한 화재 피해액이 약 62조원**에 이른다고 밝힘
- 이런 근거들로 사회적 소외계층이 안전을 위한 화재 예측 감지 AIoT 플랫폼 AI'story를 사용한다면 **재산 피해 및 인명 피해를 줄일 수 있을 것**이라고 기대된다.

경제적 기대효과 기사자료

지난 3년간 68건의 대형화재로 103명이 숨지는 등 모두 585명의 인명 피해와 1조3000억 원의 재산피해가 발생했지만, 절반에 가까운 32건은 화재 원인 조차 규명되지 못한 것으로 나타났다.

국회 행정안전위 소속 더불어민주당 이형석 의원 (광주북을)이 소방청으로부터 제출받은 자료를 검토한 결과, 2020년부터 2022년까지 총 68건의 대형 화재 발생으로 사망자 103명, 부상자 482명으로 집계됐으며 총 1조 2998억의 재산피해가 발생한 것으로 나타났다.

이처럼 대형화재로 인한 인명 및 재산 피해가 심각하지만 대형화재의 절반 가까운 47.1% 가 원인이 밝혀지지 않았다.

대형화재 원인별 현황에 따르면, 3년간 68건의 대형화재 중 원인미상 화재가 32건으로 가장 많았으며 전기적 요인으로 인한 화재가 11건, 부주의로 인한 화재가 10건으로 집계됐다. 한편, 원인 미상의 대형화재는 20년 5건, 21년 6건에서 22년 21건으로 큰 폭으로 증가했다.

이 의원은 “대형화재의 경우 많은 인명피해와 재산피해로 이어지는데도, 그 원인을 밝혀내지 못한 사례가 큰 폭으로 증가하고 있다”고 지적하며 “보다 철저한 조사로 대형화재의 원인을 규명하고 이를 통해 대형화재 예방대책을 마련할 필요가 있다”고 강조했다.

소방청에 따르면 지난해 총 4만113건의 화재가 발생해 2664명의 인명피해(사망 341, 부상 2323명)와 약 1조2070억원의 재산피해가 발생했다. 이 재산피해 금액은 전년 피해 추정 금액 약 64조1천억원에서 62조9천억원가량이 경감된 금액이다. 약 53배 이상의 피해를 줄인 셈이다.

지난해 주요 화재 피해 경감 사례로는 2월 아산시 공장 화재와 6월 천안시 학교 강당 화재를 꼽았다. 신속한 출동과 대원들의 연소 확대 저지로 각각 약 30억원, 25억원의 경제적 손실을 막았다.

김조일 119대응국장은 “국민의 신속한 119 신고와 소방기관의 총력 대응 등 민·관의 노력이 함께 했기에 가능했던 일”이라며 “앞으로도 자율적인 소방교육·훈련 시행과 소방대의 긴급출동을 위한 양보 문화 정착 등 국민의 적극적인 동참과 협조를 당부드린다”고 말했다.

실현 가능성

- 불의 온도가 화재로 번질만큼 높지 않지만, 발열량의 이상치가 감지되면 사용자에게 **발열량 이상 알림을 제공**함.
- 발열량이 높고 스파크가 있으면 인공지능이 바로 **화재 신고를 함과 동시에 주소지를 소방청에 제공**함.
- 인공지능이 학습을 하고 필요한 데이터를 센서로부터 전달받으면 **화재 위험도를 계산하여 %로 어플리케이션에 제공**함.
- 인공지능을 통해 화재 위험률을 제공하고 빠르게 신고한다는데 있어 실현 가능성이 충분하다.

활용분야

- 제품이 상용화된 후 제품성을 인정받게 되면 신도시 또는 재개발 지역의 아파트의 전 세대에 제품을 설치할 수 있다.
-> 아파트 전체에 설치하게 될 경우 어플리케이션 뿐만 아니라 월패드에서도 인공지능이 예측한 화재 위험도를 확인할 수 있도록 활용 가능하다.
- 지역별 전기적 화재 빈도수를 통계로 낼 수 있으며 이에 사용된 데이터들도 빈도수가 높은 지역은 이유를 추측하는데 근거가 될 수 있다.
-> 어느 지역에서 특별히 어떤 이유로 화재가 더 많이 발생하는지 등 데이터 분석으로 얻어낸 많은 정보를 활용하여 다양한 지리적, 환경적 특성이나 그에 맞는 예방 대책을 세우는데 활용 가능하다.

05

확장성 및 개선 방향

초기 집중 부분

- 처음 타겟층은 소외계층이 집약된 판자촌, 빈민가였음.
- 프로젝트 완성 후 사용자 타겟층의 범위를 넓혀 조사하게 됨
-> 장애인들이 화재가 났을 때 대피하는게 쉽지 않다는 사실을 알게 됨.



조사 중 발견한 다른 사례

- 2022년 8월 발생한 화재로 인해 사망한 시각장애인 50대 여성
이 거주하던 주택에는 자동화재감지탐지설비 및 스프링클러 의
무 설치 대상이 아니어 설치 되지 않았다는 기사를 보게됨.
- 소방청이 지원하는 '119 안심콜' 서비스는 지원 대상이 한정 되
어있을 뿐더러 숨짐 여성은 등록 조차 되어있지 않음

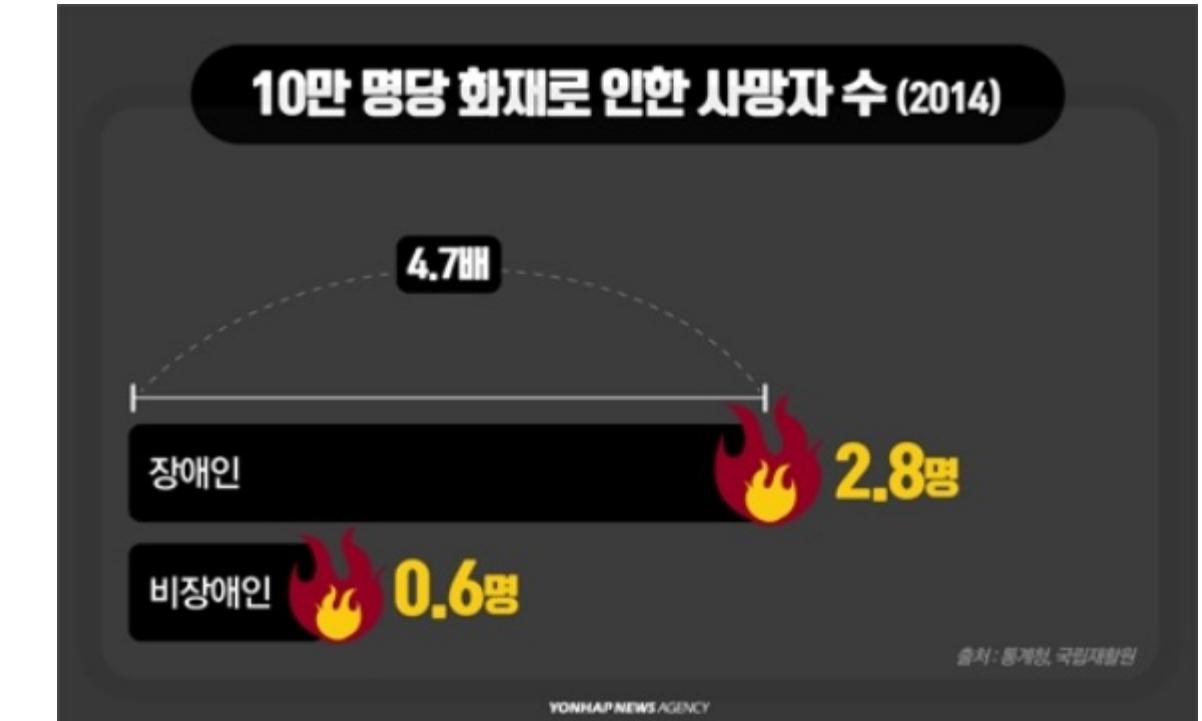
화재로 대피 못한 시각장애인 50대 여성 숨
져
해당 주택 스프링클러 등 화재감지기 설치
안 돼
소방청 '119 안심콜' 당사자 직접 등록해야
"정부 차원 종합 긴급 구조 시스템 갖춰야"

확장성 및 개선 방향

- 장애인들의 경우 대피하는데 어려움이 있어 비장애인에 비해 사망자 수가 **4배 높다**는 사실을 확인

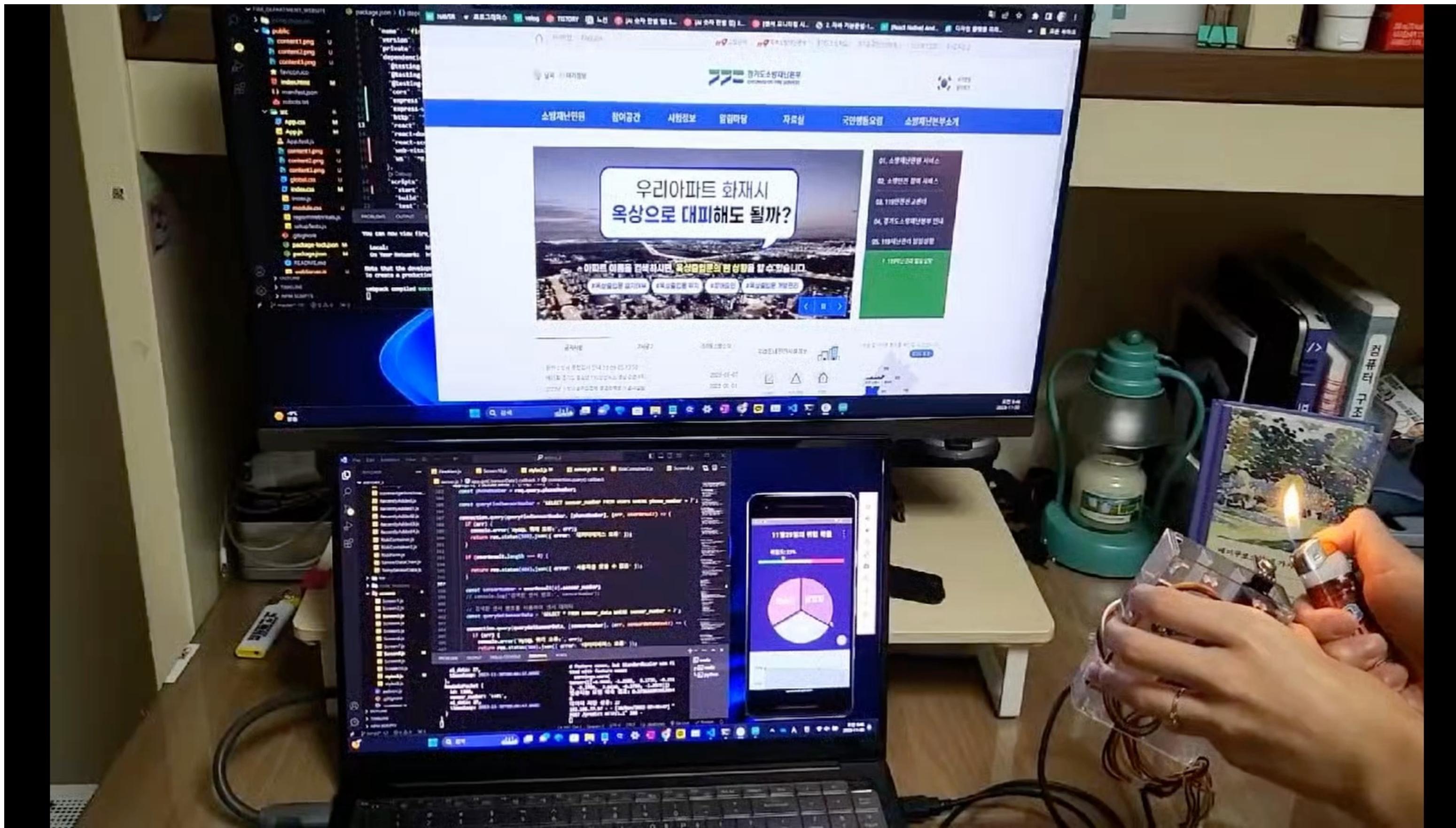


- 어플리케이션에 **거주중인 장애인이 있는지 여부를 체크**할 수 있는 기능 추가
-> 화재 발생시 신고와 동시에 **장애인 여부와 정보가 전달**되게 하여 그에 맞는 대처를 할 수 있는 방향으로 확장할 예정
- 현재 센서 가까이 반응이 있어야 감지가 잘 됨.
-> 센서를 바꿔보면서 어떤 센서가 멀리서도 화재와 스파크가 잘 감지되는지 확인해보고 어떤 거리에서 측정하는게 실내에 설치했을 때 가장 가성비가 좋은지 여러 시나리오를 만들어 개선할 예정



06

시연영상



THANK YOU
